Building a Chain-of-Thought Reasoning Module for Advanced Question Answering: An o1-Inspired Approach

Shayan Chowdhury
sc4040@columbia.edu

Priscilla Zhu
qz2531@columbia.edu

Colin Calvetti
cc4918@columbia.edu

Abstract

Users familiar with Large Language Models (LLMs), such as OpenAI's ChatGPT or Anthropic's Claude, may perceive these models as more similar to linguists than mathematicians: they perform much better on language-related tasks, such as translation or summarization, than on solving complex, multi-step mathematical or reasoning problems. Sometimes, GPT provides correct reasoning steps but ends with an incorrect final answer, while other times its "thinking" skips steps entirely and misses the mark. These limitations stem from the autoregressive nature of LLMs, whose seemingly impressive abilities are fundamentally based on predicting the next token in a sequence. Although methods like attention have been developed to help LLMs understand more complex relationships between tokens, they still lack robust logical reasoning abilities. Inspired by OpenAI's o1 series of models, which demonstrate enhanced reasoning through chain-of-thought (CoT) prompting and reinforcement learning, we investigate improving reasoning capabilities in smaller, open-source language models using Direct Preference Optimization (DPO) on Meta's LLaMA 3.1 8B model. Our experiments on the MATH subset of AGIEval show modest improvements in zero-shot settings (2 percentage points) but mixed results in few-shot scenarios, suggesting that while DPO can yield incremental gains, more fundamental architectural innovations may be necessary for substantial advances in mathematical reasoning. These findings contribute to our understanding of both the potential and limitations of current approaches to enhancing reasoning in open-source LLMs.

1 Introduction and Related Works

Current large language models (LLMs) face fundamental limitations in complex reasoning tasks due to their autoregressive nature, which focuses on predicting the next most probable token rather than true reasoning. These models are constrained by their greedy, sequential processing nature and lack the ability to backtrack or explore multiple solution paths (Borazjanizadeh and Piantadosi, 2024).

Chain-of-thought (CoT) prompting is a technique introduced by the Google Brain team to improve the reasoning abilities of large language models (Wei et al., 2022). This technique utilizes the in-context few-shot learning ability of language models and prompts them to generate intermediate reasoning steps that ultimately lead to the final answer. Doing so allows the language model to follow natural language rationales and capture logical dependencies between reasoning steps rather than jumping to a conclusion about the problem's answer. The study found that chain-of-thought prompting significantly outperforms standard prompting and enhances the language model's reasoning ability as the model scale increases.

Since then, various studies have been conducted to further improve language models' reasoning abilities through chain-of-thought prompting. In their 2024 study, Borazjanizadeh and Piantadosi (2024) proposed a neurosymbolic approach that prompts language models to extract information from problems as logical code statements, using Prolog - a logic programming language - to perform iterative deductive reasoning and arrive at the final answer. This method utilizes LLMs' understanding of linguistic patterns to analyze natural language, while leveraging Prolog's robust reasoning capabilities to infer logical solutions. The approach significantly improves LLMs' performance on the mathematical reasoning benchmark dataset GSM8K.

Chen et al. (2024) introduced the concept of "counterfactual simulatability" to evaluate how well language models can explain their own decision-making processes. Their work demonstrated that while large language models (LLMs) are trained to imitate human explanations, they often produce explanations with low precision that don't accurately reflect their internal reasoning process. In the domain of mathematical reasoning, Wang et al. (2024) explored self-training approaches enhanced with Direct Preference Optimization (DPO) to improve chain-of-thought reasoning in smaller language models. Their research showed that models trained with DPO objectives generate higher quality outputs that benefit multi-turn self-training, achieving strong performance on mathematical reasoning benchmarks like GSM8K without relying on larger models for knowledge distillation. The application of preference optimization techniques has emerged as a promising direction for enhancing model reasoning.

Furthermore, on September 12, 2024, OpenAI (2024) released their groundbreaking o1 models, introducing a "think before you respond" approach through chain-of-thought (CoT) reasoning and reinforcement learning (RL) to tackle this challenge, developing sophisticated reasoning patterns and allowing the model to spend more time processing complex problems before generating a response.

Inspired by these approaches, for our final project, we propose to build a much more distilled version of a specialized reasoning module that enhances the complex question-answering capabilities of smaller large language models (LLMs). While OpenAI maintains limited transparency about their specific technical implementations due to strategic interests, we can draw some inspiration from their "Let's Verify Step by Step" paper implementing a process supervision approach that verifies each step of reasoning in a CoT before proceeding to the next, allowing the model to catch and correct errors early in the reasoning process, before generation (Lightman et al., 2023).

Like o1, our system is designed to take time to process complex problems, break them down into manageable steps, and refine its approach through iterative reasoning before providing a final answer. Due to time constraints, while we likely won't be able to implement the entire process supervision process by Lightman et al. (2023) to evaluate each step of reasoning independently, we are interested in experimenting with how we can fine-tune a model to generate better chains-of-thought (CoTs) in a single inference call.

A key challenge in improving language model reasoning has been the development of efficient training methods that don't require extensive computational resources. Hu et al. (2022) has shown that techniques like QLoRA (Quantized Low-Rank Adaptation) enable effective fine-tuning of smaller models while maintaining reasonable memory requirements. We intend to implement DPO with QLoRA optimizations to enhance reasoning capabilities in more accessible open-source models rather than relying solely on large proprietary systems. Our goal is to demonstrate that even smaller language models can achieve significant improvements in mathematical problem-solving, though careful attention must be paid to maintaining inference speed and efficiency.

Our primary research hypothesis is that effective CoT generation can be achieved in a single inference call through careful fine-tuning, potentially offering a more computationally efficient alternative to iterative approaches.

2 Methodology

2.1 Benchmark Dataset

Since the main NLP task we are attempting to tackle is question-answering via logical reasoning, our methodology centers on utilizing the AGIEval benchmark, a comprehensive human-centric benchmark designed to evaluate foundation models across various cognitive tasks, with particular emphasis on mathematical reasoning (Zhong et al., 2023).

Specifically, we use the MATH task from AGIEval, which contains questions from high school mathematics competitions such as the American Mathematics Competitions (AMC) and the American Invitational Mathematics Examination (AIME). These questions cover a broad range of mathematical concepts, including algebra, geometry, precalculus, and number theory, often requiring complex reasoning and occasionally creativity to solve for the correct answer. Evaluating LLMs on this dataset allows us to assess their ability to reason through problems and generate novel solutions.

Table 1 provides example questions from the dataset, and an overview of the distribution of the dataset is shown in Figure 1. The dataset exhibits a diverse range of mathematical domains, with a relatively balanced distribution across each domain, ensuring comprehensive evaluation of mathematical reasoning capabilities across different difficulty levels and subject areas.

Utilizing AGIEval, we aim to create a robust dataset of reasoning examples through our pipeline.

Question	**Type**
Ben is climbing a tree with a lot of branches. His height off the ground at time t is $2t^2 - 5t + 29$ feet. To the nearest foot, what will his minimum height be?	Algebra
For how many integer values of k do the graphs of $x^2 + y^2 = k^2$ and $xy = k$ not intersect?	Intermediate Algebra
A line intersects the yz-plane at $(0, -2, -5)$, and the xz-plane at $(3, 0, -1)$. Find the point where the line intersects the xy-plane.	Precalculus
How many ways are there to fill in the tens digit and hundreds digit of $1\_\_4$ so that it is divisible by 11?	Number Theory
Suppose that $*(n) = \{n-2, n+2, 2n, \frac{n}{2}\}$. For example, $*(6) = \{4, 8, 12, 3\}$. For how many distinct integers n does $*(n)$ have exactly three distinct elements?	Counting and Probability
In a regular polygon, the measure of an interior angle is 6.5 times the measure of an exterior angle. How many sides does the polygon have?	Geometry

Table 1: AGIEval MATH dataset questions. Each question is paired with a correct answer and a detailed solution outlining the step-by-step process used to derive the result.

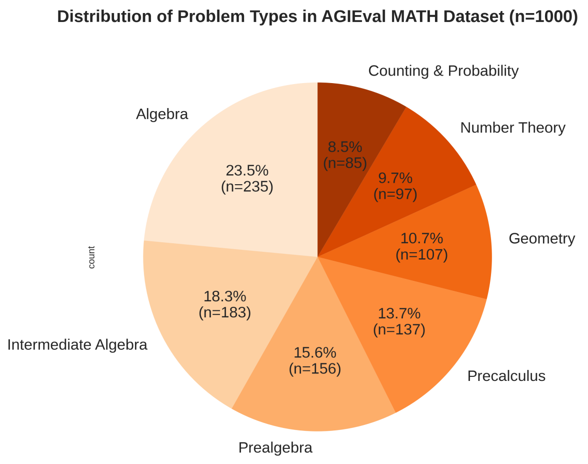


Figure 1: Distribution of mathematical problem types in the AGIEval benchmark dataset (n=1000).

2.2 Model Selection

In response to feedback on computational constraints from the one-pager, we have shifted focus to smaller models, focusing on using Meta's instruction-tuned model LLaMA-3.1-8B (Dubey et al., 2024). We also chose to replace our grader model from OpenAI's GPT-4o to Google's Gemini 1.5 Pro (Team et al., 2024) due to their offering of a free tier, with comparable grading performance.

LLMs have shown promising capabilities as judges for language model-generated content, reaching over 80% agreement with crowd-sourced human preferences in the study by Zheng et al. (2023). They showed the potential of LLM-as-a-judge while addressing bias and limited reasoning capabilities through the use of MT-Bench and Chatbot Arena. MT-Bench is designed to test the model on questions from 8 common categories to test multi-turn conversation and instruction following capabilities. Chatbot Arena has collected votes providing valuable data on the users preferred response of two LM's.

DeepMind's technical report includes examples that demonstrate the use of Gemini as a judge. Gemini's ability to translate text was compared against other Models like ChatGPT-4 Turbo by Gemini 1.0 Pro as a Judge Zheng et al. (2023).

Using an evaluation prompt with the question and a reference solution, Gemini grades the answer and reasoning provided by LLaMa. The model returns a score and boolean, evaluating the quality of the reasoning and judging the correctness of the answer by LLaMa. The graded CoT's show that Gemini is an effective judge as the more extraneous or missing steps there are in LLaMa's reasoning, the worse the assigned score is.

2.3 DPO Preference Dataset Generation Pipeline

Our pipeline consists of three main phases, followed by an inference pipeline. Figure 2 demonstrates our pipeline procedure.

To construct our dataset, we sample questions from the AGIEval MATH dataset and use predefined prompt templates to generate chain-of-thought (CoT) reasoning and corresponding answer pairs for each question. For this process, we primarily used Meta's LLaMA 3.1 8B model. The generated CoTs were then evaluated against the correct answers and solutions provided by AGIEval, with quality scores assigned using Gemini 1.5 Pro. Each CoT received a quality score between 0 and 1. Gemini 1.5 Pro uses a context window of 128,000 up to 1 million tokens. This procedure was repeated 2 times, with the temperature set at 0.7, allowing the model to explore variations in its generations in each iteration.

The quality scores from our grader LLM are used to create structured preference pairs. For each question, we identify CoTs with significantly different quality scores and pair them, with the higher-scoring CoT serving as the preferred response and the lower-scoring CoT as the rejected response. This approach is similar to positive and negative sampling in Word2Vec or the selection of "chosen" and "rejected" responses in reinforcement learning from human feedback (RLHF). In cases where the model generated CoTs with identical quality scores, we regenerated additional CoTs until at least two with distinct scores were obtained. This creates a natural ranking system that helps the model learn to distinguish between effective and ineffective reasoning patterns.

Once we've got this dataset, the goal is to fine-tune the model so it produces higher-scoring CoTs when given a question and avoids the ones we want to reject. This is achieved by optimizing a cross-entropy loss function, which measures the difference between the generated CoTs and the high-scoring CoTs in our labeled dataset.

During inference, the pipeline begins with a problem requiring complex reasoning as input. The model generates multiple candidate CoTs, each of which is scored by our grader model. The chain with the highest score is then selected to generate the final answer, ensuring the output reflects the best reasoning path.

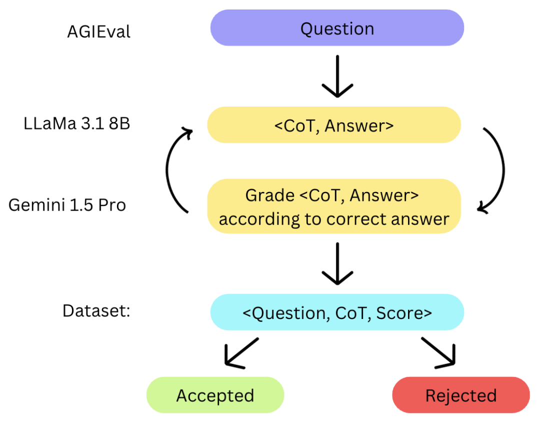


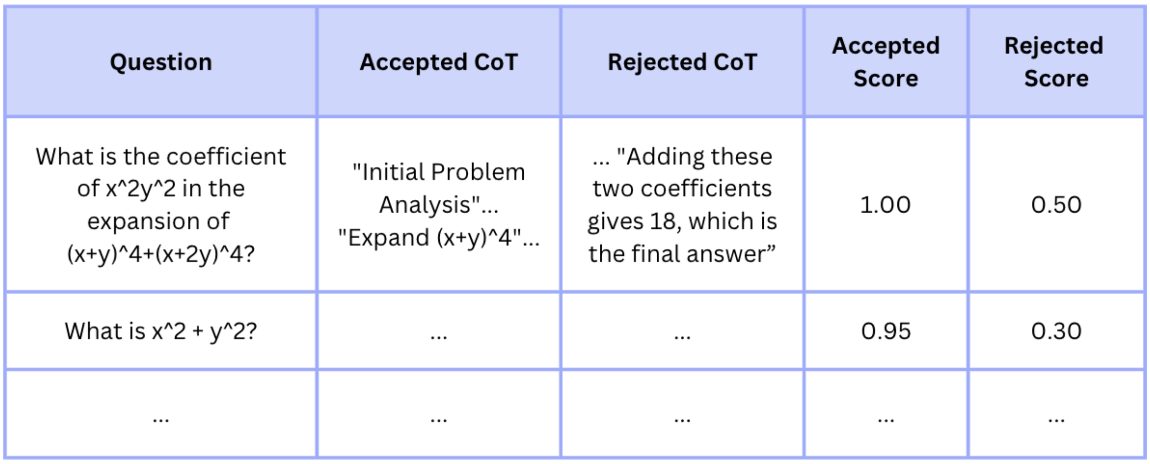
Figure 2: Pipeline visualization. The final dataset with accepted and rejected labels of CoT enters the inference pipeline of DPO.

2.4 Implementation Details

We implemented the models using the `unsloth` library (Han et al., 2023), allowing for LoRA (Low-Rank Adaptation) for faster training and inference and reduced VRAM usage.

For CoT generation, we utilize `unsloth`'s `FastLanguageModel` for inference optimization, which provides native 2x faster inference speeds compared to standard implementations. Since we are using instruction-tuned models, we reformatted the instructions, inputs, and outputs to a chat-style format for improved task performance. While we initially intended to use Stanford NLP's `DSPy` library (Khattab et al., 2022, 2023) for optimizing LLM prompting without manual prompt engineering, this template-based approach replaced our initial `DSPy` implementation due to compatibility constraints with `unsloth`'s optimization features. The generation process employs controlled parameters to ensure comprehensive yet bounded reasoning chains.

For quality assessment, we integrated a structured evaluation protocol using `DSPy` employing Gemini Pro 1.5 as the grader LLM. This grading system considers the original question posed, the CoT and final answer provided by the LLM, and the reference dataset's true solution's reasoning and answer. Finally, it outputs a normalized score between 0-1 reflecting reasoning and answer quality, along with detailed reasoning about the assessment. This scoring mechanism is crucial for our fine-tuning pipeline in, providing quantitative feedback for model optimization. The final dataset is formatted as in Figure 4.



Question	Accepted CoT	Rejected CoT	Accepted Score	Rejected Score
What is the coefficient of x^2y^2 in the expansion of (x+y)^4+(x+2y)^4?	"Initial Problem Analysis"... "Expand (x+y)^4"...	... "Adding these two coefficients gives 18, which is the final answer"	1.00	0.50
What is x^2 + y^2?	...	...	0.95	0.30
...	...	...	...	...

Figure 3: Visualization of the generated dataset with 5 columns and entries corresponding to each question.

2.5 DPO Training Implementation

We included Direct Preference Optimization (DPO) to train models for generating higher-quality chains of thought in a single inference pass (Rafailov et al., 2024). The process begins with our generated dataset, which are transformed into preference pairs suitable for DPO training.

The DPO training process implements a preference learning objective that directly optimizes the model to generate preferred outputs without explicit reward modeling. We adapted `unsloth`'s fine-tuning optimizations to utilize gradient checkpointing to reduce memory requirements during backward passes, enabling training on more modest GPU hardware while maintaining model performance.

Since our dataset has a relatively large sample size (n=1000), we included regular evaluation checkpoints where we assess the model's performance on a held-out validation set (20% of the overall dataset), measuring both the preference accuracy and the absolute quality of generated CoTs. This allows us to monitor for potential overfitting and ensure the model maintains general reasoning capabilities while optimizing for our specific objectives.

Through this comprehensive fine-tuning approach, we aim to create a model that can generate high-quality chains of thought without requiring multiple inference passes or explicit step-by-step verification, significantly reducing the computational overhead typically associated with complex reasoning tasks.

3 Results

3.1 DPO Training

Our implementation of DPO to enhance CoT reasoning capabilities demonstrates promising yet variable performance improvements over the training period. 3.1 shows the training loss and reward accuracy measurements, measured through the Weights and Biases library by Biewald (2020). We tracked approximately 150 global training steps, revealing several notable patterns in the model's learning trajectory.

Notably, the training loss trajectory over the course of DPO implementation reveals a gradual downward trend, starting from approximately 0.7 and declining to around 0.6 by the end of training, though with notable fluctuations throughout the process. The initial phase (steps 0-60) demonstrates relatively stable loss values oscillating between 0.65 and 0.75, suggesting consistent early-stage learning dynamics. Beyond step 60, we observe increased volatility in the loss function, with more pronounced downward spikes reaching as low as 0.5, particularly in the latter half of training.

This declining loss pattern, despite its variability, indicates progressive optimization of the model's preference learning objectives. The increased frequency of lower loss values (0.5-0.6) in the later stages of training (steps 100-150) aligns with the model's improving ability to generate preferred reasoning patterns. However, the persistent oscillations in loss values, even during later training stages, suggest that the optimization landscape for CoT reasoning remains challenging to navigate consistently. This volatility may reflect the complex

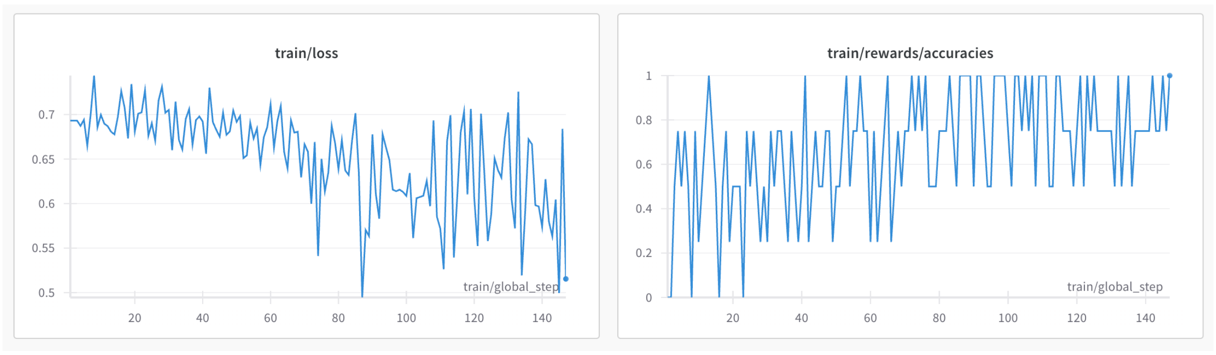


Figure 4: Training curves of Direct Preference Optimization (DPO) for improved chain-of-thought reasoning. Left: Training loss shows a gradual decline (lower is better). Right: Reward accuracies on held-out evaluation set demonstrate progressive improvement, with more frequent achievement of perfect scores (1.0) and higher baseline performance in later training stages.

nature of learning to emulate human-like reasoning processes across diverse problem contexts.

On the accuracy side (right) the training dynamics exhibit volatility as well, with accuracy values fluctuating between 0.2 and 1.0 throughout the process. While this volatility is particularly pronounced in the initial training phase (steps 0-60), a clear trend toward improvement emerges in the latter half of training (steps 80-150). This phase is characterized by more frequent achievement of perfect accuracy scores (1.0) and notably higher baseline performance. The sustained periods of high accuracy (>0.75) become more common, indicating the model's increasing ability to consistently generate reasoning patterns that align with the preferences noted in our generated dataset.

The oscillating pattern of the accuracy metrics, even in later training stages, also continues to reflect the inherent complexity of optimizing CoT generation. Further discussion on why this may be the case can be found in Section 4.

3.2 Performance Comparison to Baseline

Model	# Correct	Accuracy (%)	Δ
Base 0-shot	20/50	40.0	-
DPO 0-shot	21/50	**42.0**	+2.0
Base 2-shot	19/50	38.0	-
DPO 2-shot	19/50	38.0	0.0

For evaluation, we compared our DPO model to the base LLaMA model across both zero-shot and few-shot (2-shot) settings on a subset of 50 mathematical reasoning tasks from the MATH dataset. The results shown in 3.2 reveal interesting patterns in model performance and the impact of direct preference optimization.

Our DPO model demonstrated modest improvements in the zero-shot setting, achieving an accuracy of 42.0% compared to the base model's 40.0%. This represents a 2.0 percentage point improvement, suggesting that DPO can enhance the model's inherent reasoning capabilities without requiring exemplars. However, further testing would be needed to confirm the generalizability of these results since we only tested on a small subset and due to the time constraints of the project, we were unable to conduct the test multiple times.

In the few-shot (2-shot) setting, both models achieved identical performance with an accuracy of 38.0%. Interestingly, this represents a slight degradation from the zero-shot performance, indicating that additional examples did not necessarily lead to better reasoning outcomes in this context. One possible explanation for this is that the model may have overfitted to the few-shot examples, especially if those examples contained different types of problems and thought processes to the given held-out problem in the test set.

It's also worth noting that there is a performance ceiling observed in our experiments: 42% accuracy. This aligns with the fundamental challenges outlined in our research regarding the inherent limitations of current LLM architectures in handling complex, multi-step mathematical problems. These results suggest that while optimization techniques like DPO can yield incremental improvements, more fundamental architectural innovations may be necessary to achieve substantial advances in mathematical reasoning capabilities.

4 Conclusion

Our mixed results (marginal improvement in QA performance and oscillating patterns of loss and accuracy) in improving CoT generation through DPO reflect the inherent complexity of optimizing high-level reasoning performance in smaller, open-source language models. Unlike more straightforward preference learning tasks, reasoning capability optimization appears to maintain some degree of instability even as overall performance improves. This could be attributed to the diverse nature of reasoning patterns and the challenge of consistently applying learned preferences across varied problem contexts.

Moreover, our dataset also primarily focused on mathematical problems, as mathematical reasoning represents a particularly challenging aspect of high-level reasoning, especially without access to a calculator. However, further work is needed to evaluate the model's reasoning abilities across other logical tasks to provide a more comprehensive assessment.

Our results suggest that while DPO can marginally enhance a model's chain-of-thought reasoning capabilities, the optimization process is non-linear and potentially sensitive to the complexity of reasoning patterns being learned. The improvement in baseline performance and frequency of high-accuracy generations indicates successful preference learning, despite the maintained volatility in performance metrics.

4.1 Preference Optimization Algorithm

While we initially started with DPO as a recommendation from the initial feedback received in our project proposal, further research suggests that Proximal Policy Optimization (PPO) by Schulman et al. (2017)—another reinforcement learning algorithm may potentially offer superior performance for this specific task. A very recent study (quite literally from last week's NeurIPS 2024 conference) by Ivison et al. (2024) have shown that PPO outperforms DPO across various datasets, with particularly notable improvements in mathematical domains - showing up to 2.5% better performance. The primary advantage of PPO for mathematical reasoning tasks lies in its ability to handle complex, multi-step evaluations through iterative refinement. PPO's reinforcement learning framework is particularly well-suited for tasks requiring step-by-step verification, as it can provide granular feedback at each reasoning stage rather than just the final output. Since our grader LLM also provides specific reasoning for its score of 0-1, this feedback is crucial to incorporate with PPO for targeted alignment of the model instead of just broad preference optimization. This would likely be particularly relevant for mathematical reasoning where slight deviations in intermediate steps can lead to significantly different final results—and specific feedback on each intermediate step may help the model perform better.

Looking toward future work, we propose several key directions for improvement. First, developing a specialized reward model given the feedback from our grader LLM to evaluate both the correctness of mathematical operations and the clarity of reasoning steps would enhance the quality of the feedback signal to train our RL model. Second, incorporating advantage normalization and larger batch sizes—which is contingent on hardware, unfortunately—which have been shown to be critical factors for PPO's optimal performance in RLHF applications, could significantly improve training stability and outcomes. These proposed improvements could potentially address the current limitations of our DPO implementation and lead to more robust mathematical (and beyond) reasoning capabilities.

5 Limitations

5.1 Development Environment

Unfortunately, we faced some technical challenges during our implementation phase that constrained the scope and depth of the analysis.

We deeply appreciate Professor Muresan for providing us access to the Columbia NLP machines, but a significant limitation we faced was in configuring software dependencies, particularly PyTorch, transformers, and Unsloth, due to compatibility issues with the computing environment on the "kingcrab" machine. Despite consisting of several NVIDIA V100 GPUs, we were unfortunately unable to leverage them within the time constraits we had, since it contained multiple CUDA installations, with the default set to CUDA 8.0—an outdated version incompatible with the requirements of modern PyTorch installations. Newer versions of CUDA 11.6 and 12.1 seemed to be available on the machine, but they did not seem to be compiled, which resulted in us not being able to install newer versions of Pytorch with CUDA support.

As a result, we were unable to install certain dependencies, such as xformers—a library developed by Facebook Research for optimizing attention mechanisms in LLMs. This library is necessary for training larger models efficiently, but due to the aforementioned CUDA compatibility issues, we were unable to install it and use it for training our models. We would like to acknowledge and sincerely thank TA Nick Deas for his help in debugging these issues, such as by manually updating environment variables—but unfortunately, due to time constraints, these potential workarounds were not fully explored.

To address these challenges, our project relied on other workarounds, such as using smaller-scale models or leveraging cloud-based platforms like Google Colab. While these strategies allowed for partial implementation, they introduced constraints on the amount of data we could process for our dataset generation and evaluation pipeline—potentially limiting the generalizability of the findings. For future work, we hope to mitigate such issues by resolving the development environment issues earlier in the project or leveraging managed cloud computing resources like Google Cloud Platform.

5.2 Lack of Generalizability

Due to limitations in time and computational resources, we restricted our dataset to 50 example questions from the AGIEval MATH dataset. While our training results demonstrate promising trends, including decreasing loss and improving accuracy, the study would benefit from utilizing a larger dataset and more computational power to enhance the robustness of the findings.

Looking ahead, other tasks in AGIEval, such as LogiQA and LSAT, are exciting avenues for a more comprehensive and multifaceted evaluation of LLM reasoning performance. These tasks include logical reasoning questions that require deductive analysis and law school admission exam questions that test analytical and logical reasoning.

6 Team Member Contributions

- **Shayan Chowdhury:** Pipeline ideation and implementation, coding for DPO training, methods (DPO) writeup, results + evaluations, future work ideation w/ PPO
- **Priscilla Zhu**: Abstract, dataset generation code and graphs, methodology writeup, related work, generalizability discussion
- **Colin Calvetti**: Pipeline code, model selection writeup, LaTeX formatting, ideas for training on hardware

Acknowledgments

A massive thank you to Professor Muresan and the TAs for their time this semester and consideration. Thank you to Professor Muresan for offering access to the Columbia NLP machines for support with this project, and a special thank you to our TA Nick Deas for his help in debugging various issues with the development environment, and for his very helpful comments on our previous reports. We eagerly look forward to hearing your feedback and thoughts to continue this work further! Wishing you all happy holidays and Happy New Year!

References

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

N. Borazjanizadeh and S. T. Piantadosi. 2024. Reliable reasoning beyond natural language. *arXiv preprint arXiv:2407.11373*.

Yanda Chen, Ruiqi Zhong, Narutatsu Ri, Chen Zhao, He He, Jacob Steinhardt, Zhou Yu, and Kathleen Mckeown. 2024. Do models explain themselves? Counterfactual simulatability of natural language explanations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 7880–7904. PMLR.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daniel Han et al. 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. 2024. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *Preprint*, arXiv:2406.09279.

O. Khattab, K. Santhanam, X. L. Li, D. Hall, P. Liang, C. Potts, and M. Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*.

O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, and C. Potts. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, and K. Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

OpenAI. 2024. Learning to reason with llms.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Tianduo Wang, Shichen Li, and Wei Lu. 2024. Self-training with direct preference optimization improves chain-of-thought reasoning. *arXiv preprint arXiv:2407.18248*. StatNLP Research Group, Singapore University of Technology and Design, Soochow University.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. `https://doi.org/10.48550/arXiv.2201.11903`. ArXiv preprint arXiv:2201.11903.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, and N. Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

A Experiment Details

A.1 Inference Prompt

```
You are an expert AI assistant with
advanced reasoning capabilities. Given
a problem, your task is to provide
detailed, step-by-step explanations of
your thought process to solve it.
###
Here are some examples:
Problem: {}
Reasoning: {}
Problem: {}
Reasoning: {}
...
###
Given the following problem, please
provide a detailed, step-by-step
explanation of your thought process to
solve it:
Problem: {}
Reasoning:
```

A.2 Link to Full Code for Dataset Generation, Inference, DPO Training, etc.

```
For now in a Google Colab notebook. Will
be organized and compiled into a GitHub
repository for the future:
Link to Google Colab
```